Theoretical Computer Science 601 (2015) 67–72

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Generating square-free words efficiently

Arseny M. Shur [1]

Ural Federal University, Ekaterinburg, Russia

ARTICLE INFO

Article history:
Received 15 January 2014
Accepted 21 October 2014
Available online 14 July 2015

Keywords:
Square-free word
Random word
String algorithms

ABSTRACT

We study a simple algorithm generating square-free words from a random source. The source produces uniformly distributed random letters from a k-ary alphabet, and the algorithm outputs a $(k+1)$-ary square-free word. We are interested in the "conversion ratio" between the lengths of the input random word and the output square-free word. For any $k \geq 3$ we prove the expected value of this ratio to be a constant and calculate it up to an $O(1/k^5)$ term. For the extremal case of ternary square-free words, we suggest this ratio to have a constant expectation as well and conjecture its actual value from computer experiments.

1. Introduction

Square-free words are one of quite popular objects of study in combinatorics. Their first appearance dates back to the Thue paper [8], who proved the infiniteness of the set of ternary square-free words. Most of the time, square-free words are considered in a broader context of repetition-free words; nevertheless, they usually serve as a "firing ground" for new concepts and types of problems.

The idea of repetition-free words generated from some random source is rather new. Grytczuk, Kozik, and Witkowski used this idea to prove an avoidability result about some sort of "strong" square-freeness [2], while Camungol and Rampersad applied the same technique to get a similar result about "approximate" square-freeness [1]. The method used in these papers is basically as follows: one builds a word avoiding the desired repetitions letter by letter, choosing the letters uniformly at random from the given alphabet. At the moment when the obtained word w encounters a forbidden repetition, one dismisses some suffix of w and continues the random process. If for each n it can be proved that the word under construction reaches the length n with a nonzero probability, then the considered repetition is avoidable over the alphabet. As the authors mention explicitly, this method is inspired by the constructive proof of the Lovász local lemma, given by Moser and Tardos [3]. We should note that the techniques based on this proof already lead to several nice results in different branches of combinatorics. Speaking about combinatorics on words, we point out the solution given by Ochem and Pinlou [4] to the well-known open problem on pattern avoidability.

In this paper, we study the efficiency of this method applied to "usual" square-free words. That is, given an alphabet, we want to find the expected number of rounds of the random process needed to build a square-free word of length n. We know that the number of $(k+1)$-ary square-free words grows exponentially, and the growth rate is given by the asymptotic formula $k - 1/k - 1/k^3 + O(1/k^5)$ [6]. This formula approximates the actual growth rate very well for $k \geq 3$ and is quite reasonable even for $k = 2$ (for more information on the growth of repetition-free languages see [7]). From this formula,

E-mail address: Arseny.Shur@urfu.ru.

[1] Supported by Ministry of Education and Science of the Russian Federation under the Agreement 02.A03.21.0006 of 27.08.2013 with Ural Federal University, and by the grant 13-01-00852 of Russian Foundation for Basic Research.

http://dx.doi.org/10.1016/j.tcs.2015.07.027
0304-3975/

it seems quite probable that the expected number of rounds of the random process is only slightly over n. We introduce Algorithm R2F (Random-t(w)o-free) which implements a modification of the general method: we use random letters from a k-letter alphabet to generate a $(k+1)$-ary square-free word. Our main result is the following

Theorem 1. *The expected number of random k-ary letters used by Algorithm R2F to construct a $(k+1)$-ary square-free word of length n is*

$$N = n(1 + 2/k^2 + 1/k^3 + 4/k^4 + O(1/k^5)) + O(1). \tag{1}$$

The text is organized as follows. After short preliminaries, we introduce Algorithm R2F in Section 2. Then in Section 3 the performance of this algorithm is analyzed and Theorem 1 is proved. Section 4 contains results of computer experiments and a short discussion.

2. Preliminaries and the algorithm

We study words over the finite alphabets $\Sigma_k = \{1, \dots, k\}$, $k \geq 2$. For words, we use the array notation: $w = w[1..n]$, $w[i]$ is the ith letter of w, $w[i..j]$ is the factor of w occupying the indicated range of positions. We write $|w|$ for the length of w and λ for the empty word. A word of the form ww is a *square*; it is an *r-square* if $|w| = r$. A word is *square-free* if it contains no squares as factors.

As usual, we use the notation X^* [respectively, X^+] for the [positive] iteration of the word or language X. Studying the structure of a word, we write "$w = xy^*z$" rather than "$w \in xy^*z$".

The *growth rate* of a language $L \in \Sigma^*$ is given by the limit $\limsup_{n\to\infty} |L \cap \Sigma^n|^{1/n}$. For the languages closed under factors, lim sup can be replaced by lim.

In his WORDS'2013 lecture, Rampersad described the following algorithm to construct k-ary square-free words (the same algorithm was used in [1] to build words avoiding approximate squares). Starting with an empty word, one appends to its end one letter per round; the letter is given by a uniform random source. If the current word ends with an r-square, then one dismisses the right half of this square. The algorithm works until the constructed word reaches the required length n. In this paper, we use a modification of this algorithm; this modification generates square-free words a bit faster and is easier to analyze.

For a word $w \in \Sigma_k$, its *hash* χ_w is the permutation of Σ_k defined by "recency" of letters. Namely, a precedes b in χ_w if the rightmost position of a in w is to the right of the rightmost position of b in w. The letters that do not occur in w stay in the end of the hash in increasing order:

$$w = 136263163 \in \Sigma_6 \text{ has the hash } \chi_w = 361245.$$

Now let w have no factors aa for $a \in \Sigma_k$. Then $w[i+1] \neq w[i] = \chi_{w[1..i]}[1]$ for any $i \geq 1$. Hence w can be encoded by its first letter and a word $u \in \Sigma_{k-1}^{|w|-1}$ by the rule $u[i] = j$ if and only if $w[i+1] = \chi_{w[1..i]}[j+1]$:

$$w = 136263163 \in \Sigma_6 \text{ is encoded by } w[1] = 1 \text{ and } u = 25312322 \in \Sigma_5.$$

We use this encoding to generate a square-free word by the following

Algorithm R2F. *Input*: integers $k, n > 1$.
Output: a $(k+1)$-ary square-free word w of length n.

1. Initialization:
   - choose $w[1] \in \Sigma_{k+1}$ uniformly at random;
   - set χ_w to $w[1]$ followed by all other letters of Σ_{k+1} in increasing order;
   - set the number N of iterations to 0.
2. Append:
   - choose $j \in \Sigma_k$ uniformly at random;
   - append $a = \chi_w[j+1]$ to the end of w;
   - update χ_w shifting the first j elements to the right and setting $\chi_w[1] = a$;
   - increment N by 1.
3. Cut:
   - if w ends with an r-square, delete the last r letters of w.
4. Check for termination:
   - if $|w| < n$ then goto step 2 else return w.

Remark 1.

(1) On termination, Algorithm R2F returns a square-free word (if w ends with a square, we proceed to a shorter word which is square-free).

(2) If after step 2 w ends with a square, then this square is unique. Otherwise, it is easy to see that w contains a square inside, which is impossible because w without the last letter must be square-free.
(3) The hash is computed correctly. Indeed, when we append a letter, it moves to the first place, and the relative order of other letters remains unchanged. The deletion in step 3 does not affect the hash at all, because any two words vx and vxx have the same hash.
(4) Every $(k+1)$-ary square-free word can be the output of Algorithm R2F, because on each step we can append any letter except for the last letter of w.

Thus, Algorithm R2F converts a random k-ary word of length N to a square-free $(k+1)$-ary word of length n. Our aim is to find the expected asymptotic ratio between N and n. For convenience, we denote by U and W the final values of the random word and the obtained square-free word; the notation u and w refers to a current iteration considered (thus, u is a prefix of U).

3. Analysis

Let R_i be the number of squares of period i that appeared during the run of the algorithm. Then $n = N - \sum_{i=2}^{N/2} iR_i$. In order to evaluate n, we need to find the expectations for the numbers R_i. To do this, we describe the factors of u that produce squares during the work of the algorithm. Quite obviously, we begin with the simplest and most important case of 2-squares (1-squares are excluded by the encoding procedure).

Proposition 1. *Appending a letter to w on the current iteration produces a 2-square if and only if $u = 1(11)^+$, or $u = u'd(11)^+$ for some $u' \in \Sigma_k^*$, $d \in \Sigma_k$, $d > 1$.*

Proof. On appearance of a 2-square, the word w looks like $w'abab$; both rightmost a and b appeared when processing 1's in u. If these 1's are not consecutive, then u ends with $1y1$, and after processing the last letter of y a deletion occurred in step 3, reducing w to $w'aba$. Hence, before this deletion w ended by aba as well, implying that y ends with 1. Thus, u has the suffix 11.

If $w = ab$ or $w = w'cab$, $w' \in \Sigma_{k+1}^*$, $c \in \Sigma_{k+1}\setminus\{a,b\}$, then a 2-square cannot appear in the next iteration, but will appear in two iterations if the upcoming letters in u are 11. This observation gives us the required form of u: any first letter of u can lead to $w = ab$, while $d > 1$ is required to provide the suffix cab of w. The number of the intermediate factors 11 is irrelevant, because they leave w unchanged. □

Proposition 2. $E(R_2) = \frac{N - O(1)}{k(k+1)}$.

Proof. Let us calculate the probability of the event "the factor of the form $d(11)^t$ begins in a given position i of U", where d is any letter except 1, and $t > 0$ is fixed. If $i + 2t < |U|$, this probability is $\frac{k-1}{k^{2t+1}}$, otherwise it is zero. Such a factor causes exactly t 2-squares in w. Hence, the expected number of 2-squares caused by the factor starting in the ith position can be expressed by[2]

$$\sum_{t=1}^{\lfloor (|U|-i)/2 \rfloor} \frac{k-1}{k^{2t+1}}. \tag{2}$$

Since we are interested in the asymptotics, we neglect the borderline effects and calculate the sum (2) as the geometric series to get $\frac{1}{k(k+1)}$. By linearity of the expectation, we get $\frac{N-O(1)}{k(k+1)}$ as the expected number of 2-squares caused by all factors of the form $d(11)^t$ in U. The O-term[3] comfortably covers the approximation error as well as the 2-squares generated by the possible prefix $1(11)^t$ of U. □

Next we study 3-squares and estimate R_3.

Proposition 3. *Appending a letter to w on the current iteration produces a 3-square if and only if u has one of the following forms:*

$$u = u'd((11)^*2(11)^*2(11)^*2)^+ \quad \text{where } u' \in \Sigma_k^*, d \in \Sigma_k, d > 2; \tag{3a}$$
$$u = u'd(11)^*12((11)^*2(11)^*2(11)^*2)^+ \quad \text{where } u' \in \Sigma_k^*, d \in \Sigma_k, d > 1; \tag{3b}$$
$$u = d(11)^*2((11)^*2(11)^*2(11)^*2)^+ \quad \text{where } d \in \Sigma_k; \tag{3c}$$
$$u = (11)^+2((11)^*2(11)^*2(11)^*2)^+. \tag{3d}$$

[2] Note that we consider a nested sequence of events, and moving to the next event means exactly one more 2-square. That is why we get a sum, not a weighted sum.
[3] In fact, $O(1)$ can be replaced by $2 + O(1/k)$, but such precision is redundant here.

Proof. Let w have a suffix of the form $cabc$. If there was no deletion during the current iteration, then the last letter of u is 2. Otherwise, the period of the deleted square is not 3, because w could not have the suffix $cabcab$ after the previous step. If this period equals 2, then u ends with 11 (see Proposition 1); if the period is greater than 3, then the deleted suffix ends with $cabc$, implying that u ends with 2. Thus, w has a suffix of the form $cabc$ if and only if u has the suffix $2(11)^*$. As a result, in the moment when a 3-square is detected ($w = w'abcabc$), $u = u'2(11)^*2(11)^*2$. It remains to check which suffixes are valid for u'.

After the iteration in which the last letter of u' was processed, one has $w = w'abc$. Note that the last letter of w', if any, is distinct from c. If there was no deletion on this iteration, then $u'[|u'|] \neq 1$. Moreover, if $u'[|u'|] = 2$, then either

$$w' = \lambda,\ u' = d(11)^*2,\ d \in \Sigma_k \text{ (see (3c))},$$
$$\text{or } w' = b,\ u' = d(11)^*12,\ d \in \Sigma_k \text{ (see (3b), (3d))},$$
$$\text{or } w' = w''cb,\ u' = u''d(11)^*12,\ d \in \Sigma_k\backslash\{1\} \text{ (see (3b))}.$$

Otherwise, $u'[|u'|] > 2$ (see (3a)). If a t-square was deleted on this iteration, then u' may end by 11 (2-square) or by $2(11)^*2(11)^*2$ (3-square); if $t > 3$, then the last four letters of w before and after deletion are the same, so we apply the same argument as for the case of no deletion. □

To calculate R_3, we make use of two combinatorial lemmas.

Lemma 1 *(Folklore). The number of nonnegative solutions to the diophantine equation $x_1 + \cdots + x_s = m$ is equal to $\binom{m+s-1}{m}$.*

Lemma 2. *For any real number α, $|\alpha| < 1$, and any integer $i > 0$, the following equation holds:*

$$\sum_{n=0}^{\infty} \binom{n+i}{n} \alpha^n = \frac{1}{(1-\alpha)^{i+1}}.$$

Proof.

$$\sum_{n=0}^{\infty} \binom{n+i}{n} \alpha^n = \frac{1}{i!} \sum_{n=0}^{\infty} (n+i)\cdots(n+1)\alpha^n = \frac{1}{i!}\Big(\sum_{n=0}^{\infty} \alpha^{n+i}\Big)^{(i)}$$
$$= \frac{1}{i!}\Big(\sum_{n=0}^{\infty} \alpha^n\Big)^{(i)} = \frac{1}{i!}\Big(\frac{1}{1-\alpha}\Big)^{(i)} = \frac{1}{(1-\alpha)^{i+1}}. \qquad \square$$

Proposition 4. $E(R_3) = \frac{k^2(N - O(1))}{(k+1)(k^4+k^3-k^2-k+1)}$.

Proof. As in the proof of Proposition 2, we do not consider the factors which can appear only in the beginning of U (see (3c), (3d)) and replace geometric sequences by geometric series. These simplifications cause an aggregate error which is covered by the $O(1)$ term in the numerator.

First, let us calculate the probability P_1 of the event that U contains the factor $(11)^*2(11)^*2(11)^*2$ starting in a fixed position. Each particular factor of length $2m+3$, $m \geq 0$, has the probability $\frac{1}{k^{2m+3}}$. The number of such factors is the number of ways to distribute m factors 11 between three slots. This number equals the number of nonnegative integer solutions to the equation $x_1 + x_2 + x_3 = m$. By Lemma 1, this is $\binom{m+2}{m}$. Summing the probabilities of all these factors and applying Lemma 2 for $\alpha = 1/k^2$, we obtain

$$P_1 = \sum_{m=0}^{\infty} \binom{m+2}{m} \frac{1}{k^{2m+3}} = \frac{1}{k^3} \cdot \frac{1}{(1-1/k^2)^3} = \frac{k^3}{(k^2-1)^3}.$$

For the factors of the form $((11)^*2(11)^*2(11)^*2)^t$, where t is fixed, we calculate the probabilities P_t in the same way, applying Lemma 1 to the equation $x_1 + \cdots + x_{3t} = m$ and using Lemma 2 as above:

$$P_t = \sum_{m=0}^{\infty} \binom{m+3t-1}{m} \frac{1}{k^{2m+3t}} = \frac{1}{k^{3t}} \cdot \frac{1}{(1-1/k^2)^{3t}} = \left(\frac{k}{(k^2-1)}\right)^{3t}.$$

To count the produced 3-squares correctly, we should take into account only those factors $((11)^*2(11)^*2(11)^*2)^t$ that are preceded as indicated by (3a), (3b). The probability of being preceded by some $d > 2$ is $\frac{k-2}{k}$; the probability to be preceded by $d(11)^*12$, where $d > 1$, is

$$\frac{k-1}{k} \cdot \frac{1}{k^2} \sum_{t=0}^{\infty} \frac{1}{k^{2t}} = \frac{1}{k(k+1)}.$$

Thus, the total probability of correct precedence is $\frac{k^2-k-1}{k(k+1)}$. Multiplying it by the sum of all probabilities P_t (see the footnote on p. 69), we get the expected number of 3-squares per position:

$$\frac{k^2-k-1}{k(k+1)} \cdot \sum_{t=1}^{\infty} \left(\frac{k}{(k^2-1)} \right)^{3t} = \frac{k^2-k-1}{k(k+1)} \cdot \frac{k^3}{(k^2-1)^3 - k^3}$$
$$= \frac{k^2}{(k+1)(k^4+k^3-k^2-k+1)},$$

whence the result. □

For the number of 4-squares, we give only a rough estimation sufficient for the proof of Theorem 1.

Proposition 5. $E(R_4) = (N - O(1))(1/k^4 + O(1/k^5))$.

Proof. There are three types of 4-squares: $(abcd)^2$, $(abac)^2$, and $(caba)^2$. The first one appears when u ends with 3333, preceded by some $d > 2$. The appearance of such a factor in a fixed position has the probability $1/k^4 - O(1/k^5)$. Adding the probabilities of longer factors by encoding short squares between 3's or getting several 4-squares in the same position affects only the terms covered by $O(1/k^5)$ (even by $O(1/k^6)$, to be more precise). To define each of the remaining two types of 4-squares, one needs to fix five letters in u: for $(abac)^2$, u should end with $1d1212$, where $d > 2$; for $(caba)^2$, u should end with $d'd12121$, where $d, d' > 1$. Thus, the expectation of the number of these squares in a fixed position are covered by $O(1/k^5)$. The result for $E(R_4)$ now follows. □

We cannot extend the proof of Proposition 5 to r-squares for any $r \geq 5$, because the number of such squares can be big with respect to k. Instead, we estimate the number of r-squares for $r \geq 5$ using the results and techniques on the growth rates of repetition-free languages. As we mentioned in the introduction, the growth rate of the $(k+1)$-ary square-free language equals $k - 1/k - 1/k^3 - O(1/k^5)$. This estimate was obtained in [6] by closing the gap between the upper and the lower bounds to $O(1/k^5)$. For the upper bounds, the square-free languages were replaced by bigger regular languages. These languages are called m-approximations and defined by finite sets of forbidden factors; such sets consist of all r-squares with $r \leq m$. For lower bounds, the result of [5] was used: if α_m is the growth rate of the m-approximation of the target language, then any number γ such that $\gamma + \frac{1}{\gamma^{m-1}(\gamma-1)} < \alpha_m$ is less than the growth rate of the target language. So, for $m = 5$ the gap between the upper and the lower bounds shrinks to $O(1/k^5)$.

The lower bound is based on the inequality which can be very easily reformulated in our terms: the total probability of deletion of a word of length r in a fixed step of Algorithm R2F is at most $1/\gamma^r$. But $1/\gamma^r = 1/k^r + O(1/k^{r+2})$; thus, the probabilities of deletion of long words are covered by the O-term. Adding the results of Propositions 2, 4, and 5, we finally get

$$n = (N - O(1)) \left(1 - \frac{2}{k(k+1)} - \frac{3k^2}{(k+1)(k^4+k^3-k^2-k+1)} - \frac{4}{k^4} - O\left(\frac{1}{k^5}\right) \right)$$
$$= (N - O(1)) \left(1 - \frac{2}{k^2} - \frac{1}{k^3} - O\left(\frac{1}{k^5}\right) \right). \tag{4}$$

Inverting the expression in parentheses in (4), we obtain (1). Theorem 1 is proved.

4. Results of computer experiments and discussion

We wrote a C program implementing Algorithm R2F and ran experiments, building square-free words over the alphabets of 3, 4, 5, 10, and 20 letters. For each alphabet, the total length of constructed words is over 1 000 000. The experimental data for the alphabets with more than 3 letters correlate very well with the theory. Namely, the actual numbers of r-squares for small r are 0.01–2% away of the expected values, and the contribution of long squares is quite small. The average ratio N/n is 1.406 for 4 letters; 1.171 for 5 letters; 1.0266 for 10 letters; 1.0057 for 20 letters.

Of course, the case of the ternary alphabet is the most intriguing. The general asymptotic formula is too rough for the case $k = 2$. For example, the equality $k - 1 = 1$ can lower the order of some terms; on the other hand, the contribution of long words to the sum of probabilities is quite big and can raise the order of some terms. In addition, some lengths of squares are impossible. The experimental data gives the following picture: the ratio N/n has a constant expectation (checked for $n = 3000$, $n = 10\,000$, and $n = 32\,000$), which is approximately 12.5; valuable contribution is made by all squares with periods up to 13.

The main conclusion of this paper is that simple stochastic algorithms can generate complicated combinatorial objects with high efficiency under "normal" conditions ($k \geq 3$ in our case). Moreover, such algorithms can be successfully applied in "extremal" conditions. The behaviour of Algorithm R2F in the extremal case $k = 2$ is worth a separate study.

References

[1] S. Camungol, N. Rampersad, Avoiding approximate repetitions with respect to the longest common subsequence distance, arXiv:1308.1620 [math.CO], 2013.
[2] J. Grytczuk, J. Kozik, M. Witkowski, Nonrepetitive sequences on arithmetic progressions, Electron. J. Combin. 18 (1) (2011) P209.
[3] R.A. Moser, G. Tardos, A constructive proof of the general Lovász local lemma, J. ACM 57 (2010) 11.
[4] P. Ochem, A. Pinlou, Application of entropy compression in pattern avoidance, Electron. J. Combin. 21 (2014) P2.7.
[5] A.M. Shur, Two-sided bounds for the growth rates of power-free languages, in: Proc. 13th Int. Conf. on Developments in Language Theory, DLT 2009, in: LNCS, vol. 5583, Springer, 2009, pp. 466–477.
[6] A.M. Shur, Growth of power-free languages over large alphabets, in: Proc. 5th International Computer Science Symposium in Russia, CSR 2010, in: LNCS, vol. 6072, Springer, 2010, pp. 350–361.
[7] A.M. Shur, Growth properties of power-free languages, Comput. Sci. Rev. 6 (2012) 187–208.
[8] A. Thue, Über unendliche Zeichenreihen, Nor. Vidensk. Selsk. Skr. Mat.-Nat. Kl. 7 (1906) 1–22.